```
README contains a list of files in a tree for understanding connections.

This project is not finished.

It is highly modularized, but it does not fulfill the requirements of the assignment.

To compile, type "make" and the makefile will compile with "gcc -Wall main.c -o tokenizer"

Typing "make clean" will run "rm tokenizer"

To run the program after compilation, type the following:
        ./tokenizer "sepators" "tokens"

If the wrong number of arguments are given, usage information is printed to the screen.

Again, this program does not work.

I believe the data structure is valid, and I believe that if we had prioritized a procedure for
separating the tokens (rather than just a pseudocode explaination), we would have had more success.
We do recognize however that this part is the _essential_ part of the program, and that we will recieve
very little credit.

(however, any partial credit awarded would be much appreciated)

Here are our test cases, containing examples of:
        No seperators
        No Tokens
        All escape sequences
        Trying to go over 25 characters
        Going over 25 escape sequences
        2 mixes of both
        empty arguements
        too little arguements

Input: "\"?\n\t\v\n\r\f\a\\", "what am I doing?"
Expected Output: "what", "am", "I", and "doing"

Input: "\t\v\b\n/?", "/NOMOREERROR/S/?/n/sssss"
Expected Output: "NOMOREEERROR", "S", "n" and "sssss"

Input: "\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\", "\\\\\\\\\\"
Expected Output: ""

Input: "\'?","\\'a'?ab\t\v\b"
Expected Output:"a", "ab","t","v", and "b"

Input: "", "no more''''\n"
Expected Output:"no", "more", and "n"

Input: "???????\\\\n\\\\\\\\\\\\\?\\\\\\\\\\\\\\\000", "O/ctal"
Expected Output: "O" and "ctal"

Input:"\xhh\'\f\?\"\'","b?ac?kereknreisreriuesruneruesuresubrereureureurbeurbeu"
Expected Output: "b", "ac", and "kereknreisreriuesruneruesuresubrereureureurbeurbeu"

Input:"\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n
\n\n\n\n\n\n\n\n\n\n", "\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n
\n\n\n\n\n\a"
Expected Output:"a"

Input:"",
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
Expected Output:
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA

Input:"/?/?/?/?/%/r/f/a/'/r/a/f/?/r/?", "\a\a\a\a\a\a\a\hello who's there?"
Expected Output: "a", "a","a","a","a","a","a","a", "hello", "who", "s", and "there"
```